

WHAT IS CLAIMED:

- 1 1. A method of performing high precision emulation of program code
2 instructions for a subject machine on a target machine, comprising:
3 determining if operands in instructions of the program code for the subject
4 machine require a different precision than provided for by the target machine; and
5 applying a floating point emulation algorithm to perform intermediate
6 calculations on the operands of the instructions at a higher precision than the precision
7 supported by the target machine

- 1 2. The method of claim 1, wherein the target machine includes
2 floating point hardware and integer hardware, wherein said floating point emulation
3 algorithm comprises:
4 utilizing floating point hardware on the target machine to perform
5 calculations on the operands of the instructions when it is determined based upon the
6 intermediate calculations that the target machine provides sufficient precision for the
7 calculations required by the instructions; and
8 utilizing integer hardware on the target machine to perform calculations
9 not selected to be performed by the floating point hardware

1 3. The method of claim 2, wherein the program code instructions are
2 accumulated instructions that are calculated at a higher precision than the operands
3 capable of being handled by the target machine.

1 4. The method of claim 3, wherein the program code instructions are
2 floating point accumulated instructions of the form: $d = \pm(a*b \pm c)$,
3 wherein a, b, c and d are operands expressible as floating point numbers.

1 5. The method of claim 4, further comprising identifying whether any
2 of the operands (a, b, or c) are special values having a known result that all compatible
3 hardware will produce regardless of the level of precision of said hardware.

1 6. The method of claim 5, wherein said special values include either
2 zero, infinity, or NaN (not a number), wherein the floating point hardware is utilized to
3 calculate the result of the accumulated instructions when any of the operands (a, b, or c)
4 are identified as special values.

1 7. The method of claim 4, wherein said floating point emulation
2 algorithm further comprises:
3 determining whether the exponent for the result of the multiplication of
4 $(a*b)$ overlaps with the exponent of c; and

5 utilizing the floating point hardware to calculate the result of the
6 accumulated instructions when the exponent for the result of the multiplication of ($a*b$)
7 fails to overlap with the exponent of c .

1 8. The method of claim 7, wherein, when the exponent for the result
2 of the multiplication of ($a*b$) overlaps with the exponent of c , said floating point
3 emulation algorithm further comprising:
4 determining whether the mantissa for the result of the multiplication ($a*b$)
5 requires more mantissa bits than provided for by said floating point hardware; and
6 utilizing the floating point hardware to calculate the result of the
7 accumulated instructions when the result of the multiplication ($a*b$) does not require
8 more mantissa bits than provided for by the floating point hardware.

1 9. The method of claim 8, said floating point emulation algorithm
2 further comprising computing the full calculation of $a*b$ using the integer hardware when
3 mantissa for the result of the multiplication ($a*b$) requires more mantissa bits than
4 provided for by the floating point hardware.

1 10. The method of claim 9, said floating point emulation algorithm
2 further comprising:
3 determining whether the final resulting mantissa of the mantissa($a*b$) - the
4 mantissa (c) equals zero;

5 utilizing the floating point hardware to make the result equal to zero when
6 the resulting mantissa is equal to zero; and
7 calculating the remaining parts of the calculation of $a*b + c$ using the
8 integer hardware when the final resulting mantissa is not equal to zero.

1 11. A computer-readable storage medium having software resident
2 thereon in the form of computer-readable code executable by a computer to perform the
3 following steps in the high precision emulation of program code instructions for a subject
4 machine on a target machine:

5 determining if operands in instructions of the program code for the subject
6 machine require a different precision than provided for by the target machine; and

7 applying a floating point emulation algorithm to perform intermediate
8 calculations on the operands of the instructions at a higher precision than the precision
9 supported by the target machine

1 12. The computer-readable storage medium of claim 11, wherein the
2 target machine includes floating point hardware and integer hardware, wherein said
3 floating point emulation algorithm comprises:

4 utilizing floating point hardware on the target machine to perform
5 calculations on the operands of the instructions when it is determined based upon the
6 intermediate calculations that the target machine provides sufficient precision for the
7 calculations required by the instructions; and

8 utilizing integer hardware on the target machine to perform calculations
9 not selected to be performed by the floating point hardware

1 13. The computer-readable storage medium of claim 12, wherein the
2 program code instructions are accumulated instructions that are calculated at a higher
3 precision than the operands capable of being handled by the target machine.

1 14. The computer-readable storage medium of claim 13, wherein the
2 program code instructions are floating point accumulated instructions of the form: $d =$
3 $\pm(a*b \pm c)$,
4 wherein a, b, c and d are operands expressible as floating point numbers.

1 15. The computer-readable storage medium of claim 14, said
2 computer-readable code further executable for identifying whether any of the operands
3 (a, b, or c) are special values having a known result that all compatible hardware will
4 produce regardless of the level of precision of said hardware.

1 16. The computer-readable storage medium of claim 15, wherein said
2 special values include either zero, infinity, or NaN (not a number), wherein the floating
3 point hardware is utilized to calculate the result of the accumulated instructions when any
4 of the operands (a, b, or c) are identified as special values.

1 17. The computer-readable storage medium of claim 14, wherein said
2 floating point emulation algorithm further comprises:

3 determining whether the exponent for the result of the multiplication of
4 (a*b) overlaps with the exponent of c; and

5 utilizing the floating point hardware to calculate the result of the
6 accumulated instructions when the exponent for the result of the multiplication of (a*b)
7 fails to overlap with the exponent of c.

1 18. The computer-readable storage medium of claim 17, wherein,
2 when the exponent for the result of the multiplication of (a*b) overlaps with the exponent
3 of c, said floating point emulation algorithm further comprises:

4 determining whether the mantissa for the result of the multiplication (a*b)
5 requires more mantissa bits than provided for by said floating point hardware; and

6 utilizing the floating point hardware to calculate the result of the
7 accumulated instructions when the result of the multiplication (a*b) does not require
8 more mantissa bits than provided for by the floating point hardware.

1 19. The computer-readable storage medium of claim 18, said floating
2 point emulation algorithm further comprising computing the full calculation of a*b using
3 the integer hardware when mantissa for the result of the multiplication (a*b) requires
4 more mantissa bits than provided for by the floating point hardware.

1 20. The computer-readable storage medium of claim 19, said floating
2 point emulation algorithm further comprising:
3 determining whether the final resulting mantissa of the mantissa($a*b$) - the
4 mantissa (c) equals zero;
5 utilizing the floating point hardware to make the result equal to zero when
6 the resulting mantissa is equal to zero; and
7 calculating the remaining parts of the calculation of $a*b + c$ using the
8 integer hardware when the final resulting mantissa is not equal to zero.

1 21. In combination:
2 a target processor; and
3 translator code for performing high precision emulation of program code
4 instructions for a subject machine on a target machine, said translator code comprising
5 code executable by said target processor for performing the following steps:
6 determining if operands in instructions of the program code for the subject
7 machine require a different precision than provided for by the target machine; and
8 applying a floating point emulation algorithm to perform intermediate
9 calculations on the operands of the instructions at a higher precision than the precision
10 supported by the target machine

1 22. The combination of claim 21, wherein the target machine includes
2 floating point hardware and integer hardware, wherein said floating point emulation
3 algorithm comprises:

4 utilizing floating point hardware on the target machine to perform
5 calculations on the operands of the instructions when it is determined based upon the
6 intermediate calculations that the target machine provides sufficient precision for the
7 calculations required by the instructions; and

8 utilizing integer hardware on the target machine to perform calculations
9 not selected to be performed by the floating point hardware

1 23. The combination of claim 22, wherein the program code
2 instructions are accumulated instructions that are calculated at a higher precision than the
3 operands capable of being handled by the target machine.

1 24. The combination of claim 23, wherein the program code
2 instructions are floating point accumulated instructions of the form: $d = \pm(a*b \pm c)$,
3 wherein a, b, c and d are operands expressible as floating point numbers.

1 25. The combination of claim 24, wherein said floating point
2 emulation algorithm comprises identifying whether any of the operands (a, b, or c) are
3 special values having a known result that all compatible hardware will produce regardless
4 of the level of precision of said hardware.

1 26. The combination of claim 25, wherein said special values include
2 either zero, infinity, or NaN (not a number), wherein the floating point hardware is
3 utilized to calculate the result of the accumulated instructions when any of the operands
4 (a, b, or c) are identified as special values.

1 27. The combination of claim 24, wherein said floating point
2 emulation algorithm further comprises:
3 determining whether the exponent for the result of the multiplication of
4 (a*b) overlaps with the exponent of c; and
5 utilizing the floating point hardware to calculate the result of the
6 accumulated instructions when the exponent for the result of the multiplication of (a*b)
7 fails to overlap with the exponent of c.

1 28. The combination of claim 27, wherein, when the exponent for the
2 result of the multiplication of (a*b) overlaps with the exponent of c, said floating point
3 emulation algorithm further comprises:
4 determining whether the mantissa for the result of the multiplication (a*b)
5 requires more mantissa bits than provided for by said floating point hardware; and
6 utilizing the floating point hardware to calculate the result of the
7 accumulated instructions when the result of the multiplication (a*b) does not require
8 more mantissa bits than provided for by the floating point hardware.

1 29. The combination of claim 28, said floating point emulation
2 algorithm further comprising computing the full calculation of $a*b$ using the integer
3 hardware when mantissa for the result of the multiplication ($a*b$) requires more mantissa
4 bits than provided for by the floating point hardware.

1 30. The combination of claim 29, said floating point emulation
2 algorithm further comprising:
3 determining whether the final resulting mantissa of the mantissa($a*b$) - the
4 mantissa (c) equals zero;
5 utilizing the floating point hardware to make the result equal to zero when
6 the resulting mantissa is equal to zero; and
7 calculating the remaining parts of the calculation of $a*b + c$ using the
8 integer hardware when the final resulting mantissa is not equal to zero.